Remarks

The Examiner's allowance of claims 5, 7-10 and 15-16 (page 2, ¶ 2) is noted with appreciation.

Claim 6, indicated as being allowable if rewritten in independent form (page 2, ¶ 3), has been rewritten in such form. Per a telephone interview held today with the Examiner, there is no rejection of claim 6 under 35 U.S.C. § 112, second paragraph, and the reference to such a rejection in the latest Office action is in error.

Finally, new claim 17 is based upon claim 1, but is directed to a method that basically recites the functions of the activation module.

Claims 1-4, 6, and 11-14 again stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Deianov et al. 6,529,985 ("Deianov") in view of Hammond 6,463,583, either alone (page 2, ¶ 4) or in conjunction with other art (page 5, ¶ 12; page 7, ¶ 19). These rejections are again respectfully traversed.

Claim 1, upon which the remaining claims under rejection depend, is directed to a system for intercepting API calls in a virtual memory environment. The system comprises an activation module (AM) and an interception module (IM) (Fig. 3). The interception module selectively provides modified functionality for the intercepted API calls. The activation module loads the interception module to occupy a location in a shared region of virtual memory as long as interception of the API calls is required. The activation module redirects the API calls by creating an alias to any page containing an entry point for an API call to be intercepted (page 9, lines 13-14) and writes the address of the interception module to the alias (lines 15-17). The activation module provides to any instances of the interception module the original entry points for the API calls. Thus, in the embodiment shown, the real memory location containing the DosAllocMem far32 pointer address in the module DOSCALL1 to the actual kernel module is replaced with a pointer to the interception module, which resides in a global shared region of virtual memory and performs specified preprocessing before branching to the original entry point in the kernel module (page 9, lines 1-17).

Deianov discloses a system for selective interception of API calls. It is an example of the prior art system (with its attendant disadvantages) described in the background portion of applicant's specification at page 3, lines 24-30, in which addresses in an interrupt vector table are altered to point to alternative code at the altered address. As shown in Fig. 1 of the patent, one embodiment of the system includes an interception module 111 and an interrupt vector table 113 in operating system (OS) address space 105, together with a modified loader program 121 and a system call wrapper 125 in user address space 103. (In other embodiments, the system call wrapper 125 may be in OS address space.) In the interrupt vector table 113, selected pointers 114 to system calls 115 are replaced with pointers 118 to the interception module 111.

The Examiner equates applicant's claimed interception module with Deianov's interception module 111. However, as the Examiner concedes (page 3, ¶ 7), Deianov's interception module 111 resides in OS address space 105 (Figs. 2-3) and is not loaded "to occupy a location in a shared region of virtual memory" as claimed by applicant. As a result, Deianov's interception module cannot be loaded except when the system is restarted.

The Examiner asserts, however, that Hammond teaches this feature and that it would have been obvious to combine the two references because Hammond's teaching of loading the interception module "would improve the flexibility of Deianov's system by dynamically injecting the execution logic into a shared memory space of a window[ed] operating system" (page 4, ¶ 8). Applicant respectfully disagrees.

Contrary to the Examiner's assertion, Hammond does not teach loading an interception module to occupy a location in a shared region of virtual memory, as claimed by applicant. Rather, Hammond discloses a system for dynamically injecting execution logic in the form of a dynamic link library (DLL) into a shared memory space of a windowed operating system (col. 2, lines 56-58; col. 8, line 36 to col. 9, line 3). This execution logic, however, is not an "interception module", since it does not intercept calls meant for other modules, as do applicant and Deianov Other than the fact that it loads an executable module into an address space, Hammond has nothing in common with either Deianov or applicant's claimed invention. Certainly Hammond

would not suggest loading Deianov's interception module 111 into such shared user space when that reference so repeatedly teaches that it be loaded into the OS address space 105 (e.g., col. 5, line 64 to col. 6, line 2; col. 10, lines 26-30; col. 11, lines 54-58), and when there is no indication that the interception module 111 would even work in shared user space.

Additionally, there is no teaching in either Deianov or Hammond of redirecting API calls by creating an alias to any page containing an entry point for an API call to be intercepted and writing the address of the interception module to the alias. Applicant uses this technique because the DOSCALL1 module from which API calls are redirected is a read-only module and thus cannot be directly altered (page 2, lines 12-14). By writing the address of the interception module to the alias (mapping to the same location in real memory) rather than to the DOSCALL1 module itself, applicants overcome this obstacle. Deianov, by contrast, directly modifies the pointers in the interrupt vector table 113 and does not use any alias as claimed by applicant.

Conclusion

For the foregoing reasons, claim 1 and the claims dependent thereon as well as new claim 1 are believed to distinguish patentably over the art cited by the Examiner. Reconsideration of the application as amended is therefore respectfully requested. It is hoped that upon such consideration, the Examiner will hold all claims allowable and pass the case to issue at an early date. Such action is earnestly solicited.

Respectfully submitted,
RICHARD JOHN MOORE

By 

William A. Kinnaman, Jr.
Registration No. 27,650
Phone: (845) 433-1175
Fax: (845) 432-9601

WAK/wak